Linköpings universitet
IDA Department of Computer and Information Sciences
Prof. Dr. Christoph Kessler

TENTAMEN / *EXAM*

TDDD56

Multicore and GPU Programming

29 aug 2015, 14:00–18:00, TER2

Jour: Christoph Kessler (070-3666687, 013-282406), visiting ca. 16:00.
Ingemar Ragnemalm (070-6262628), by phone only.

Hjä lpmedel / *Admitted material:*

– Engelsk ordbok / *Dictionary from English to your native language*

General instructions

- This exam has 9 assignments and 5 pages, including this one.
  Read all assignments carefully and completely before you begin.
- It is recommended that you use a new sheet of paper for each assignment, because they will be corrected by different persons.
  Sort the pages by assignment, number them consecutively and mark each one on top with your exam ID and the course code.
- You may answer in either English or Swedish. **English is preferred** because not all correcting assistants understand Swedish.
- Write clearly. Unreadable text will be ignored.
- Be precise in your statements. Unprecise formulations may lead to a reduction of points.
- Motivate clearly all statements and reasoning.
- Explain calculations and solution procedures.
- The assignments are *not* ordered according to difficulty.
- The exam is designed for 40 points. You may thus plan about 5 minutes per point.
- Grading: U, 3, 4, 5. The preliminary threshold for passing is 20 points.
- An exam review session will be announced on the course homepage.

1. (5 p.) **Multicore Architecture Concepts**

   (a) Define and explain the following technical terms:

      i. SIMD (vector) instructions
      ii. Dennard Scaling
      iii. Bus snooping
      iv. Sequential (memory) consistency
      v. False sharing

   (Remember that an example is not a definition. Be general and thorough.) (5p)

2. (5 p.) **Parallel Programming with Threads and Tasks**

   (a) What does *thread pinning* mean, and what is its purpose? (2p)

   (b) Today, mutual exclusion locks are typically implemented by hardware atomic instructions like *test-and-set*. Using *spinlocks* for mutual exclusion allows to react quickly if the lock becomes available, but has a detrimental effect on performance. Which one, and why? (1p)

   We discussed two programming techniques to deal with this problem. Describe one of them. (2p)

   (c) How does a work-stealing task scheduler work? (2p)

3. (6 p.) **Non-blocking Synchronization**

   (a) Name 2 problems of lock-based synchronization that are removed by non-blocking synchronization. (1p)

   (b) In the lecture, we considered a *fair lock* implementation using the atomic `FetchAndIncr` operation:

```
// two shared counters, statically initialized to 0:
shared int ticket = 0;  // next waiting ticket to grab
shared int active = 0;  // ticket now entitled to enter CS

void acquire()   // acquire fair lock:
{
  int myticket = FetchAndIncr( &ticket, 1 );
  while (myticket != active)
     ;  // busy waiting
}

void release()   // release fair lock:
{
  active ++;
}
```

   (This implementation assumes a sequentially consistent memory.)

   You are given a multicore processor that has no atomic *fetch_and_increment* instruction but that has a *compare_and_swap* (CAS) instruction instead. Rewrite the above fair lock implementation using CAS such that the behavior is the same. Explain your solution. (2.5p)

   (c) Can the ABA problem occur in your CAS-based implementation of the fair lock? Explain why or why not. If yes, how likely is it to occur? Motivate your answer. (2p)

   (d) A core component of task-based parallel programming environments (such as Cilk, TBB, OpenMP3, StarPU etc.) is a dynamic (user-level) task scheduler that maps tasks to worker threads at runtime. A common design option is to use a central shared task queue data structure for the dynamic task scheduler. Why could a dynamic task scheduler possibly benefit from using a non-blocking shared data structure (instead of a lock-protected one)? (Short answer) (0.5p)

4. (5 p.) **Design and Analysis of Parallel Algorithms**

   (a) Formulate Brent's Theorem (explained formula), give its interpretation, derive it (calculation), and describe its implications for parallel algorithm design and analysis. (3p)

   (b) What is a *parallel speedup anomaly*? (1p)

   Give an example of a *parallel speedup anomaly* that may occur on a multicore computer, and explain its technical cause. (1p)

[In case of questions about the following 3 assignments, ask I. Ragnemalm in the first hand.]

5. **GPU algorithms**

   Write code (or pseudo code) for applying a 5×5 low-pass filter on an image in a reasonably optimized way. The image could be any size. The code should apply the kernel below. Full score requires close to real code that takes more than one optimization technique into account. The strategy for memory usage must be clearly described. You may use CUDA or OpenCL syntax as you please. (5p)

1	4	6	4	1
4	16	24	16	4
6	24	36	24	6
4	16	24	16	4
1	4	6	4	1

6. **GPU Architecture Concepts**

   (a) Describe the major architectural differences between a multi-core CPU and a GPU (apart from the GPU being tightly coupled with image output). Focus on the differences that are important for parallel computing. (3p)

   (b) Compare shared memory, global memory and register memory in terms of performance, usage and accessibility. CUDA terminology is assumed, please note if you use OpenCL terminology. (2p)

7. **GPU Quickies**

   (a) What do you have to do for achieving better performance by coalescing? (1p)

   (b) In CUDA, you can use the modifier `__host__`. What does this signify? (1p)

   (c) Some operations can be implemented either as gather or scatter operations. Which is most suitable for a GPU implementation, and why? (1p)

   (d) In GPU Computing using the graphics pipeline, in what stage are the computations usually carried out? (1p)

   (e) Can you rely on any threads/work items in a GPU to be executed simultaneously? Which ones? (1p)

8. (3 p.) **Optimization and Parallelization**

   (a) Name and shortly describe two different loop transformations that can improve the cache hit rate of the loop, and explain why. (3p)

9. (1 p.) **Parallel algorithmic design patterns and High-level parallel programming**

   Give two main advantages and two main drawbacks of *skeleton-based parallel programming*. (1p)

Good luck!